INFO 6205 Spring 2022 Project
Menace

Team Members:

1. Bhawna Agarwal (002938098)
2. Heer Mehta (002190162)
3. Anusha Poojary (002114479)

Repository Link: https://github.com/mehtaheer/PSA_FINAL_PROJECT

• Introduction

o Aim

Implement "The Menace" by replacing matchboxes with values in a hash table(the key will be the state of the game)

Train the Menace by running games played against the "human" strategy based on optimal strategy.

One will need to choose values for:
•alpha(the number of "beads" in each "matchbox" at the start of the game—maybe different for each move: first move, second move, etc.)
•beta(the number of "beads" to add to the "matchbox" in the event of a win)
•gamma(the number of "beads" to take to the "matchbox" in the event of a loss)
•delta(the number of "beads" to add to the "matchbox" in the event of a draw)

o Approach

- We created the dictionary for setting up the beads initially with nine keys with a value of 8 each. The current state list and board states' dictionary have also been created to keep the track of states generated.
- The random method is used to choose the random probability(r) and then compare it by adding the probability(p) for each box which is 1/9 or 0.11111111. We keep summing up the number until r < p. Then, that index is chosen, and the value at that key of the current state dictionary is replaced by 1.

- Now, the process repeats until the winning or draw state is obtained.
- Each time the menace wins, it is rewarded with 3 beads, if loses then we remove 1 bead and if the game is drawn then 1 bead is added to the state. By this, we are maximizing the chance of Menace to win the game.
- We are using tuple to make the state immutable
- After the training is done, the data is stored in the “training. pickle” file
- This file is used by menace to make smart moves while playing against a human player.

• **Program**

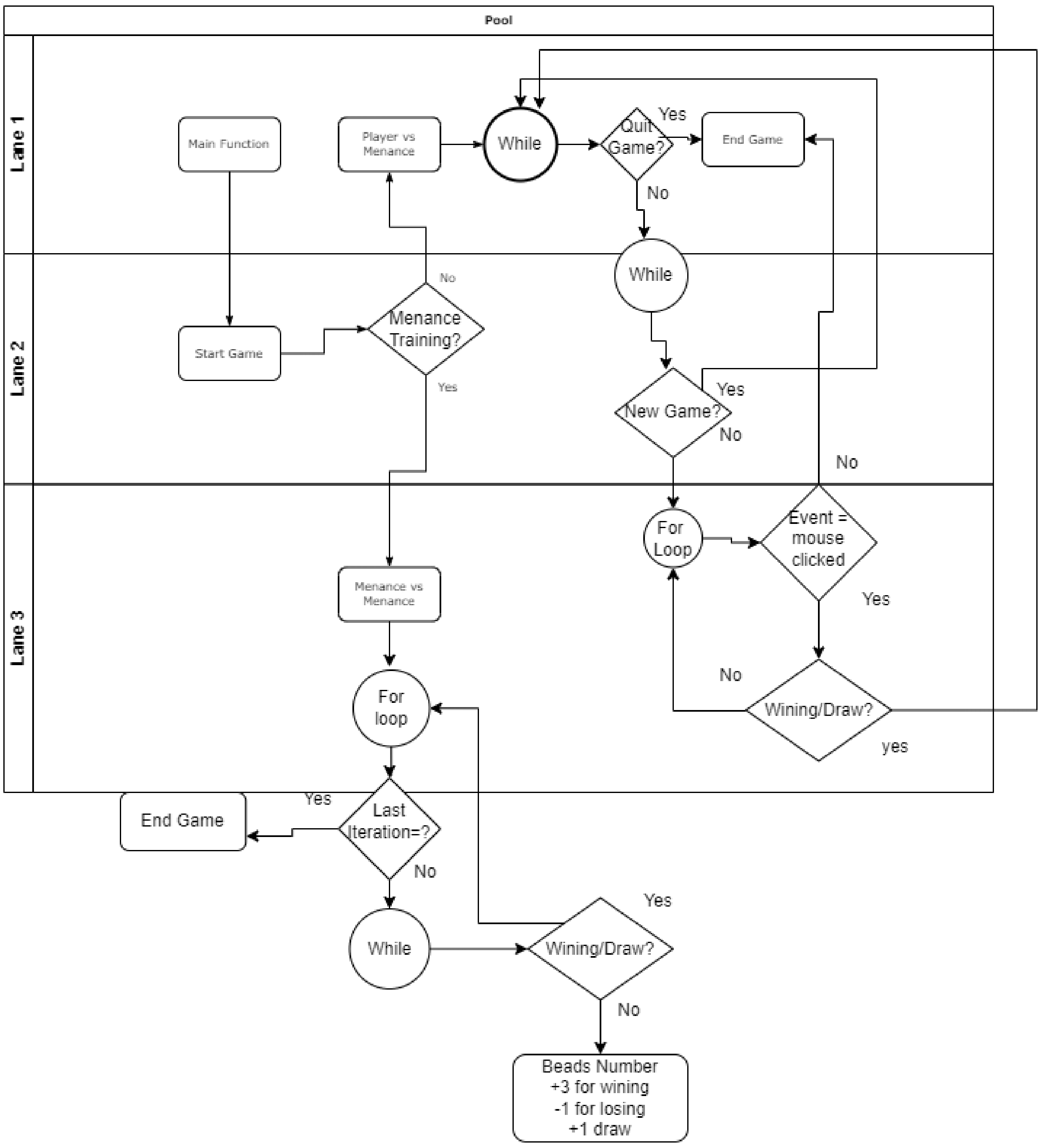
o Data Structures & Classes:

We have used List and Dictionary as Data structures for this project.

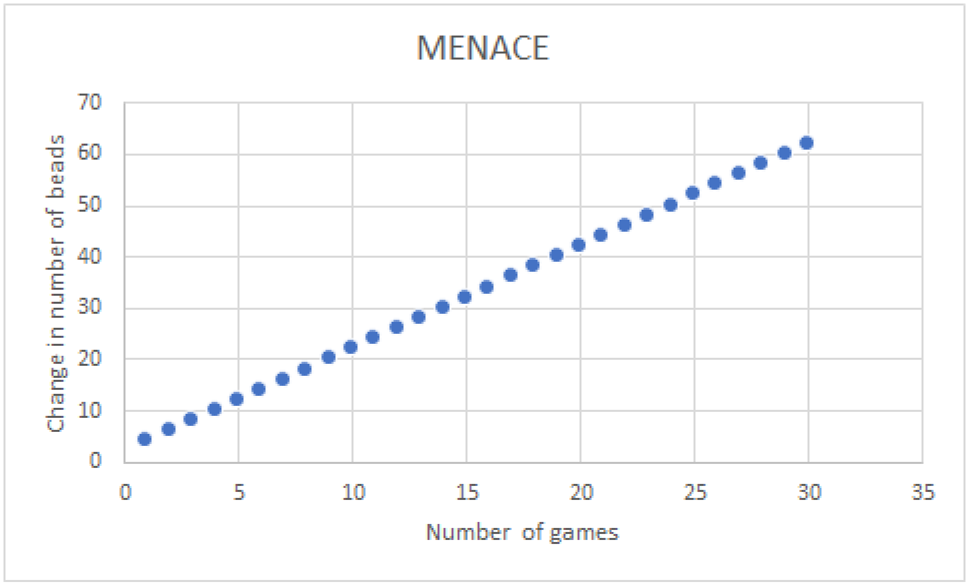
o Algorithm – Minimax algorithm is used to achieve the trained menace

o Invariants – Winning game positions are constant.

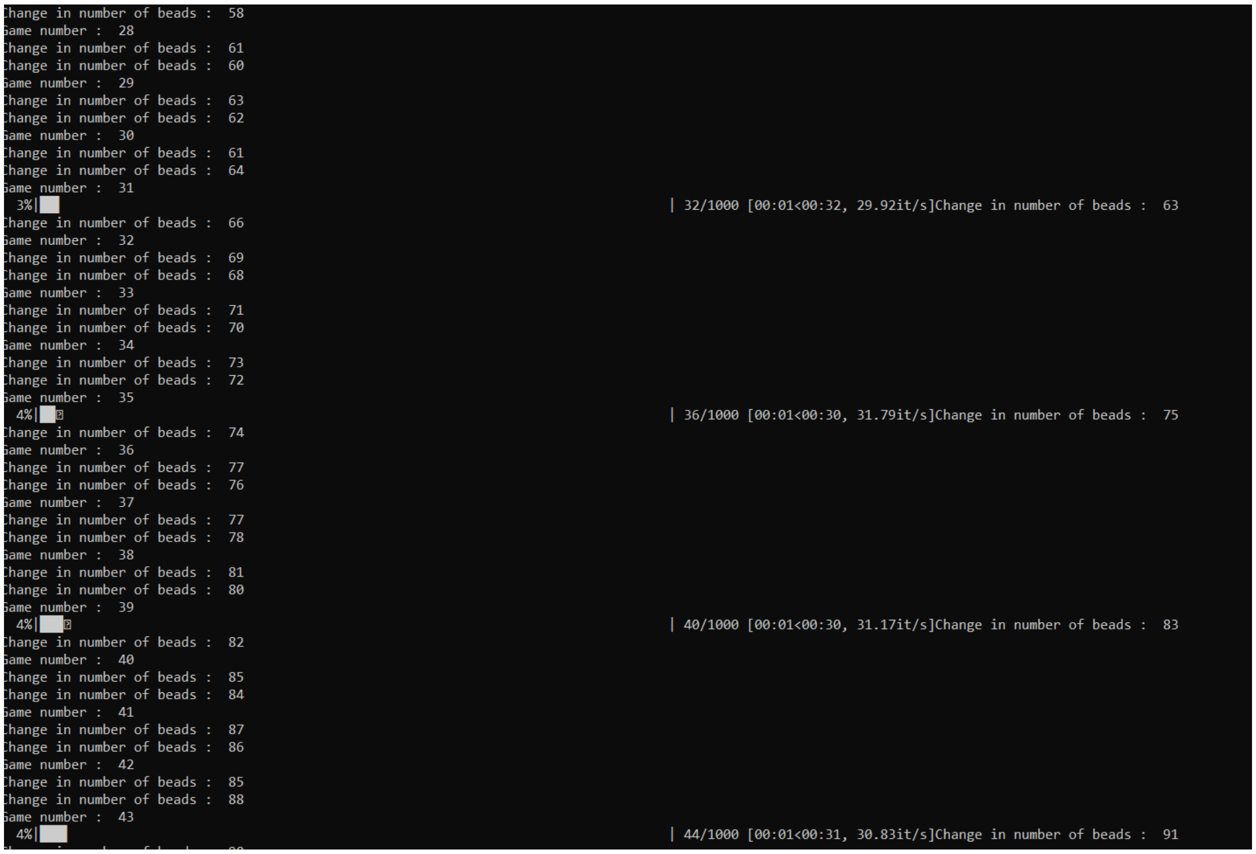
•**Flow Charts (inc. UI Flow)**



•Observations & Graphical Analysis –

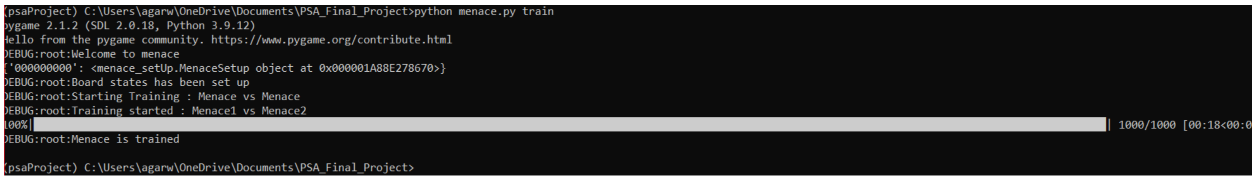


•Results & Mathematical Analysis



```
hange in number of beads :  58
ame number :  28
hange in number of beads :  61
hange in number of beads :  60
ame number :  29
hange in number of beads :  63
hange in number of beads :  62
ame number :  30
hange in number of beads :  61
hange in number of beads :  64
ame number :  31
 3%|█                                                                 | 32/1000 [00:01<00:32, 29.92it/s]Change in number of beads :  63
hange in number of beads :  66
ame number :  32
hange in number of beads :  69
hange in number of beads :  68
ame number :  33
hange in number of beads :  71
hange in number of beads :  70
ame number :  34
hange in number of beads :  73
hange in number of beads :  72
ame number :  35
 4%|█                                                                 | 36/1000 [00:01<00:30, 31.79it/s]Change in number of beads :  75
hange in number of beads :  74
ame number :  36
hange in number of beads :  77
hange in number of beads :  76
ame number :  37
hange in number of beads :  77
hange in number of beads :  78
ame number :  38
hange in number of beads :  81
hange in number of beads :  80
ame number :  39
 4%|██                                                                | 40/1000 [00:01<00:30, 31.17it/s]Change in number of beads :  83
hange in number of beads :  82
ame number :  40
hange in number of beads :  85
hange in number of beads :  84
ame number :  41
hange in number of beads :  87
hange in number of beads :  86
ame number :  42
hange in number of beads :  85
hange in number of beads :  88
ame number :  43
 4%|██                                                                | 44/1000 [00:01<00:31, 30.83it/s]Change in number of beads :  91
```

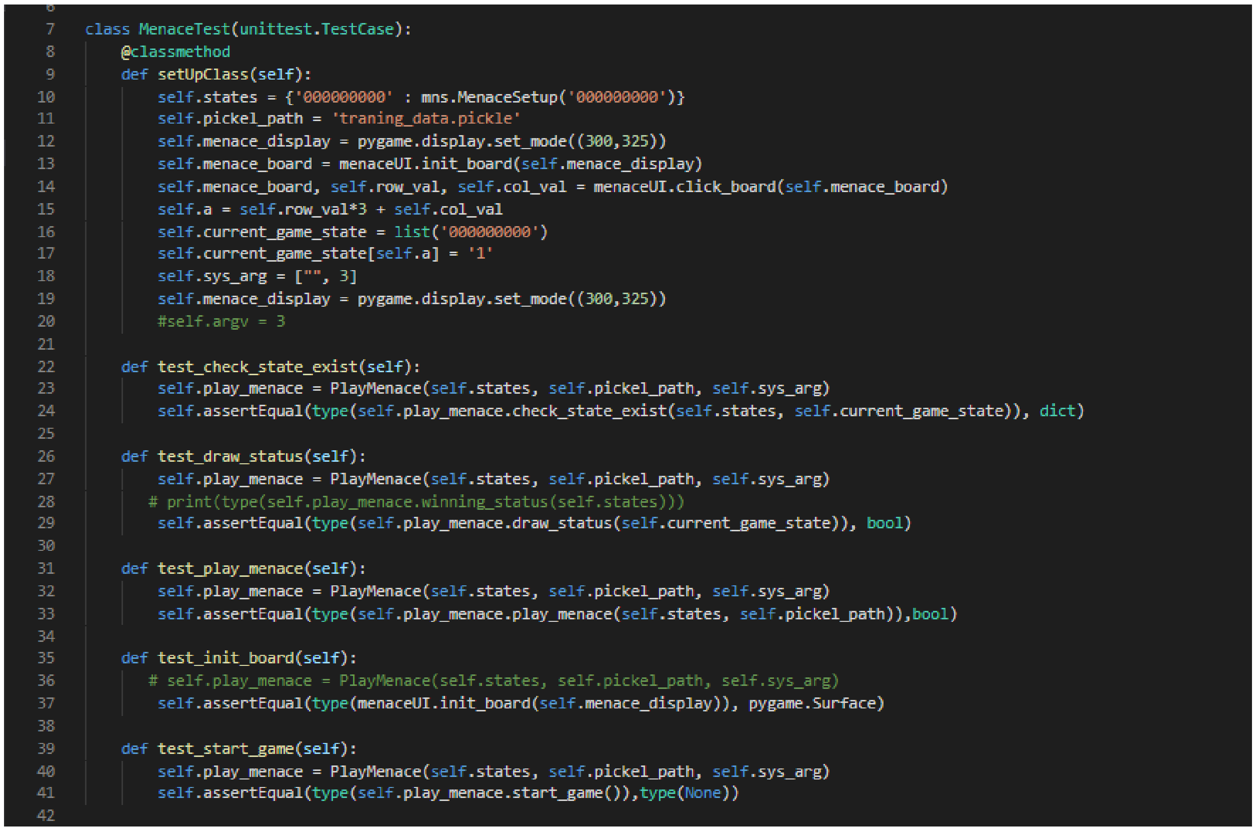
The change in number of beads is linear to number of games



```
(psaProject) C:\Users\agarw\OneDrive\Documents\PSA_Final_Project>python menace.py train
ygame 2.1.2 (SDL 2.0.18, Python 3.9.12)
ello from the pygame community. https://www.pygame.org/contribute.html
EBUG:root:Welcome to menace
'000000000': <menace_setUp.MenaceSetup object at 0x000001A88E278670>}
EBUG:root:Board states has been set up
EBUG:root:Starting Training : Menace vs Menace
EBUG:root:Training started : Menace1 vs Menace2
00%|████████████████████████████████████████████████████████████| 1000/1000 [00:18<00:0
EBUG:root:Menace is trained

(psaProject) C:\Users\agarw\OneDrive\Documents\PSA_Final_Project>
```

•Testcases



```python
class MenaceTest(unittest.TestCase):
    @classmethod
    def setUpClass(self):
        self.states = {'000000000' : mns.MenaceSetup('000000000')}
        self.pickel_path = 'traning_data.pickle'
        self.menace_display = pygame.display.set_mode((300,325))
        self.menace_board = menaceUI.init_board(self.menace_display)
        self.menace_board, self.row_val, self.col_val = menaceUI.click_board(self.menace_board)
        self.a = self.row_val*3 + self.col_val
        self.current_game_state = list('000000000')
        self.current_game_state[self.a] = '1'
        self.sys_arg = ["", 3]
        self.menace_display = pygame.display.set_mode((300,325))
        #self.argv = 3

    def test_check_state_exist(self):
        self.play_menace = PlayMenace(self.states, self.pickel_path, self.sys_arg)
        self.assertEqual(type(self.play_menace.check_state_exist(self.states, self.current_game_state)), dict)

    def test_draw_status(self):
        self.play_menace = PlayMenace(self.states, self.pickel_path, self.sys_arg)
       # print(type(self.play_menace.winning_status(self.states)))
        self.assertEqual(type(self.play_menace.draw_status(self.current_game_state)), bool)

    def test_play_menace(self):
        self.play_menace = PlayMenace(self.states, self.pickel_path, self.sys_arg)
        self.assertEqual(type(self.play_menace.play_menace(self.states, self.pickel_path)),bool)

    def test_init_board(self):
       # self.play_menace = PlayMenace(self.states, self.pickel_path, self.sys_arg)
        self.assertEqual(type(menaceUI.init_board(self.menace_display)), pygame.Surface)

    def test_start_game(self):
        self.play_menace = PlayMenace(self.states, self.pickel_path, self.sys_arg)
        self.assertEqual(type(self.play_menace.start_game()),type(None))
```

• Conclusion

The more the MENACE is trained, the better decision it takes to win the game.

First, we tried 500 games, which wasn’t sufficient to train the MENACE as it was losing the game when played against Human players. We kept increasing the number of games from 500 to1000 to 10000 to 100000 and finally concluded that our MENACE was well trained with less probability of losing games when introduced for 100000 games or more.

•References

1. Matchbox Educable Noughts and Crosses Engine-WIKIPEDIA
2. SAMPLE LESSON: Matchboxes Play Tic-Tac-Toe-YouTube Video
3. MENACE by Scroggs
4. pygame Documentation